A modeling and algorithmic framework for (non)social (co)sparse audio restoration

C. Gaultier, N. Bertin, S. Kitić, R. Gribonval,

***Abstract*—We propose a unified modeling and algorithmic framework for audio restoration problem. It encompasses analysis sparse priors as well as more classical synthesis sparse priors, and regular sparsity as well as various forms of structured sparsity embodied by shrinkage operators (such as social shrinkage). The versatility of the framework is illustrated on two restoration scenarios: denoising, and declipping. Extensive experimental results on these scenarios highlight both the speedups of 20% or even more offered by the analysis sparse prior, and the substantial declipping quality that is achievable with both the social and the plain flavor. While both flavors overall exhibit similar performance, their detailed comparison displays distinct trends depending whether declipping or denoising is considered.**

***Index Terms*—Sparsity, Time-Frequency, Structure, Denoising, Declipping**

I. Introduction and Motivations

WETHER originating from the acquisition, the analog-to-digital conversion or any other later processing step, distortion in audio signals induces unwanted effects and has a negative impact on application performance. Distortion may consist in clipping, packet loss or additive noise. The resulting degraded Signal-to-Noise ratio (SNR) directly affects speech understanding, listening comfort and automated tasks like speech recognition or signal classification. Audio enhancement aims at restoring such distorted signals. Typical approaches to signal restoration or reconstruction rely on spectral subtraction [1], autoregressive or statistical models [2], and lately, neural networks [3]. Recently, a body of work popularized the explicit formulation of the enhancement task as an *inverse problem*, and accordingly, the idea of solving it through the use of a time-frequency *sparse regularization* [4], [5], [6].

A. *Analysis* vs *Synthesis*

The *sparse synthesis model* assumes that the signal of interest $\mathbf{x}$ is built from a linear combination of atoms aggregated in a large dictionary $\mathbf{D}$. We could more precisely write

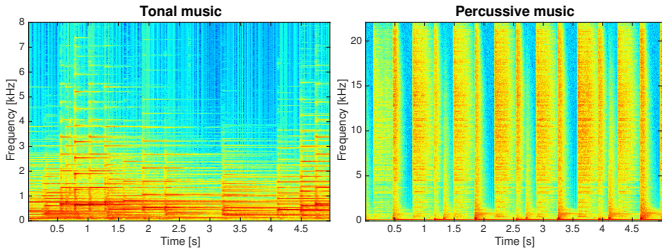
$$\mathbf{x} = \mathbf{D}\mathbf{z} \tag{1}$$

with $\mathbf{x} \in \mathbb{R}^{\mathsf{L}}$ the time domain signal, $\mathbf{D} \in \mathbb{C}^{\mathsf{L}\times\mathsf{S}}$ the dictionary and $\mathbf{z} \in \mathbb{C}^{\mathsf{S}}$ a sparse representation of the vector $\mathbf{x}$.

While such synthesis approaches comprise a vast majority of the sparsity-based time-frequency regularization techniques, it has been demonstrated recently [6], [7] that the *analysis sparse model*, which assumed name is cosparse model, can reveal more advantageous, in particular in terms of computational cost. Instead of estimating a sparse representation $\mathbf{z}$ of the signal $\mathbf{x}$ through the sparse synthesis model, the rationale of the cosparse model is to estimate the signal $\mathbf{x}$ itself assuming that

$$\mathbf{z} = \mathbf{A}\mathbf{x} \tag{2}$$

is sparse with $\mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}$ called the analysis operator. The two models are equivalent when $\mathsf{P} = \mathsf{S} = \mathsf{L}$ and $\mathbf{A}\mathbf{D} = \mathbf{I}$.



(a) Tonal music (b) Transient music

Fig. 1. Musical excerpts spectrograms

B. *Structured Sparsity*

Structured forms of sparsity such as group sparsity [8] or social sparsity [9] have emerged as useful refinements of the above mentioned techniques to take into account the typical time-frequency patterns of audio signals, as illustrated on Figures 1a and 1b. Figure 1a displays the spectrogram of a tonal musical excerpt, where high energy coefficients are structured across time reflecting the strong presence of harmonics. Figure 1b displays the spectrogram of a percussive music sample, where the dominant coefficients gather across frequency due to transients and beats.

C. *Contributions*

In this paper, we introduce a new joint modeling and algorithmic framework encompassing sparse and cosparse models as well as plain or structured sparsity for time-frequency audio restoration. The versatility of the framework is illustrated on two audio reconstruction tasks: declipping and denoising[1]. This is achieved through a unique adaptive algorithmic structure designed to encompass independently the modeling variations or the specific features of different audio enhancement tasks.

The paper is organized as follows. Section II introduces the general algorithmic framework. Section III (resp. Section IV) instantiates this framework for denoising (resp. declipping)

C. Gaultier, N. Bertin and R. Gribonval are with Univ Rennes, Inria, CNRS, IRISA. While preparing this work S. Kitić was with Technicolor R&D.

[1]Audio examples are available at https://project.inria.fr/spare/

including extensive experimental results. On both scenarios, while the analysis and synthesis flavors yield almost identical restoration performance, significant speedups are obtained with the analysis one, hence its performance is studied in more details. For declipping, it yields significant quality improvements for speech and most music styles. The performance of plain *vs* social cosparsity is often comparable but significantly distinct trends are observed between the denoising and declipping scenarii when varying the input degradation level. The final section collects last notes and suggests future insights.

D. Notations

In the following, lower-case Greek symbols (ε) stands for scalar constant. Lower-case sans serif font (i) denotes an integer. Lower-case bold font ($\mathbf{v}$) expresses a vector and upper-case ($\mathbf{V}$) a matrix. $\mathbf{v}_{\mathsf{i}}$ is an i^{th} element of a vector and $\mathbf{v}^{(\mathsf{i})}$ an i^{th} iterate. Θ is used for a set. $\mathcal{O}$ stands for a non-linear operator and F a functional. $\mathbf{V}_{(\mathsf{ij})}$ represents the component of the matrix $\mathbf{V}$ indexed the i^{th} row and j^{th} column. Finally, $\mathbf{V}^{\mathsf{H}}$ denotes the Hermitian transpose of a matrix $\mathbf{V}$. Curved relation symbols ($\preccurlyeq, \succcurlyeq, \prec, \succ$) are used for entry-wise comparisons between matrices. Any other notation will be disambiguated in the text.

II. General Framework

In this section, we present a general framework using either simple sparse modeling (analysis or synthesis based) or structured sparse priors to address reconstruction problems in audio.

A. Modeling Framework

As for many other problems in audio, we cast the problem of recovering a clean signal $\mathbf{x}$ from noisy indirect measurements $\mathbf{y}$ as a linear inverse problem, stated in a frame-based manner.

We consider the matrix $\mathbf{Y}_{\mathsf{n}}$ containing one or more windowed frames of L samples from the observed signal $\mathbf{y}$. The problem of audio reconstruction is to estimate the original clean signal frames, similarly gathered in a matrix $\mathbf{X}_{\mathsf{n}}$.

This can be expressed as an inverse problem, which is usually ill-posed, hence the need to regularize it thanks to prior knowledge on the signal to recover. Such knowledge can be to assume that the signal to recover admits a time-frequency representation endowed with some form of sparsity.

1) Analysis and synthesis sparse modeling: We consider one frame at a time, i.e. $\mathbf{X}_{\mathsf{n}} \in \mathbb{R}^{\mathsf{L}\times 1}$, and $\mathbf{Z}_{\mathsf{n}}$ an approximate frequency representation of $\mathbf{X}_{\mathsf{n}}$ assumed to have few significant coefficients. The assumed relation between $\mathbf{X}_{\mathsf{n}}$ and $\mathbf{Z}_{\mathsf{n}}$ depends on the type of sparse model:

Analysis sparse model	**Synthesis sparse model**
$\mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \geq \mathsf{L}$	$\mathbf{D} \in \mathbb{C}^{\mathsf{L}\times\mathsf{S}}, \mathsf{S} \geq \mathsf{L}$
$\mathbf{Z}_{\mathsf{n}} \simeq \mathbf{A}\mathbf{X}_{\mathsf{n}}, \mathbf{Z}_{\mathsf{n}} \in \mathbb{C}^{\mathsf{P}\times 1}$	$\mathbf{D}\mathbf{Z}_{\mathsf{n}} \simeq \mathbf{X}_{\mathsf{n}}, \mathbf{Z}_{\mathsf{n}} \in \mathbb{C}^{\mathsf{S}\times 1}$
$\|\mathbf{Z}_{\mathsf{n}}\|_0 \ll \mathsf{P}$;	$\|\mathbf{Z}_{\mathsf{n}}\|_0 \ll \mathsf{S}$.

The matrix $\mathbf{A}$ (resp. $\mathbf{D}$) embodies a forward (resp. backward) frequency transform (e.g. DCT or DFT), possibly made redundant with zero-padding.

2) From plain to structured sparse modeling: The above-described models – which will be denoted as "plain" sparse models – treat separately each frame of the signal. In contrast, "structured" sparse modeling introduces dependencies between frequency representations of adjacent time frames. For this, we consider the matrix $\mathbf{X}_{\mathsf{n}} \in \mathbb{R}^{\mathsf{L}\times(2\mathsf{b}+1)}$ which columns are the frames of the original signal indexed by $[\mathsf{n}-\mathsf{b}, \mathsf{n}+\mathsf{b}]$, and $\mathbf{Z}_{\mathsf{n}}$ a matrix which columns are frequency representations of these frames. In other words, this matrix is a *time-frequency* representation of the underlying audio signal.

In structured sparse models, the assumed relation between $\mathbf{Z}_{\mathsf{n}}$ and $\mathbf{X}_{\mathsf{n}}$ becomes:

Structured analysis model	**Structured synthesis model**
$\mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \geq \mathsf{L}$	$\mathbf{D} \in \mathbb{C}^{\mathsf{L}\times\mathsf{S}}, \mathsf{S} \geq \mathsf{L}$
$\mathbf{Z}_{\mathsf{n}} \simeq \mathbf{A}\mathbf{X}_{\mathsf{n}}, \mathbf{Z}_{\mathsf{n}} \in \mathbb{C}^{\mathsf{P}\times(2\mathsf{b}+1)}$	$\mathbf{D}\mathbf{Z}_{\mathsf{n}} \simeq \mathbf{X}_{\mathsf{n}}, \mathbf{Z}_{\mathsf{n}} \in \mathbb{C}^{\mathsf{S}\times(2\mathsf{b}+1)}$
$\|\mathbf{Z}_{\mathsf{n}}\|_0 \ll \mathsf{P}\times(2\mathsf{b}+1)$	$\|\mathbf{Z}_{\mathsf{n}}\|_0 \ll \mathsf{S}\times(2\mathsf{b}+1)$
$\mathbf{Z}_{\mathsf{n}}$ is "structured";	$\mathbf{Z}_{\mathsf{n}}$ is "structured".

In this paper, to instantiate the notion of "structured" sparse modeling, we rely on the concept of *social sparsity* [9], *i.e.* we assume that the indices of non-zero coefficients in $\mathbf{Z}_{\mathsf{n}}$ are organized according to some known time-frequency *patterns* that will be illustrated in Section II-C.

Remark: for the special case where $\mathsf{b} = 0$, both $\mathbf{X}_{\mathsf{n}}$ and $\mathbf{Z}_{\mathsf{n}}$ have a single column, and the social sparse modeling collapses to the plain sparse assumption.

B. Algorithmic Framework

Given a distorted matrix of observations $\mathbf{Y}_{\mathsf{n}}$, our goal is to find means to recover an estimate $\hat{\mathbf{X}}_{\mathsf{n}}$ of the frames $\mathbf{X}_{\mathsf{n}}$ of the original signal. For this, one seeks $\hat{\mathbf{X}}_{\mathsf{n}}$ that satisfies:

- a data fidelity constraint with respect to $\mathbf{Y}_{\mathsf{n}}$, according to some distortion model (additive noise, clipping...);
- the modeling constraints described above.

This is the spirit of the algorithmic framework we develop. It relies on two components:

- a *generalized projection* onto the data-fidelity constraint;
- a *shrinkage* enforcing (structured) sparsity.

These are combined into an iterative algorithm analog to Douglas-Rachford (DR) splitting [10]. This generic algorithmic framework can be instantiated in different ways depending on the target application. Two specific examples will be thoroughly described in Sections III (denoising) and Section IV (declipping).

The first component is the generalized projection:

Definition 1 (Generalized projection)**.** *Let Θ be a nonempty* convex *set, and $\mathbf{M}$ be a full column rank matrix. Given a time-frequency matrix $\mathbf{Z}$, we denote $\mathcal{P}_{\Theta,\mathbf{M}}(\mathbf{Z})$ the (unique) solution of the following optimization problem:*

$$\underset{\mathbf{W}\in\Theta}{\text{minimize}} \ \|\mathbf{M}\mathbf{W} - \mathbf{Z}\|_F. \tag{3}$$

The computation of this projection for some particular choices of constraint set Θ and matrix $\mathbf{M}$ will be discussed in due time.

The second component is the shrinkage operator. Intuitively, this operator gives an output which is "decreased" in a certain sense, with respect to its input argument, hence somewhat promoting sparsity. Although we will not formally exploit it for any convergence analysis, we also recall below the notion of shrinkage, also called "thresholding rule" [11].

Definition 2 (Shrinkage)**.** *$\mathcal{S}(\cdot)$, is a shrinkage if:*

1) *$\mathcal{S}(\cdot)$ is an odd function;*
2) *$0 \leq \mathcal{S}(x) \leq x$, for all $x \in \mathbb{R}^+$.*
3) *$(\mathcal{S}(\cdot))_+$ is nondecreasing on $\mathbb{R}^+$ and $\lim_{x\to+\infty}(\mathcal{S}(x))_+ = +\infty$, where $(\cdot)_+ := \max(\cdot, 0)$.*

When applied to a (time-frequency) matrix, and written $\mathcal{S}(\mathbf{Z})$, shrinkage is applied entry-wise.

In a concrete setting, the following is required to instantiate the framework:

Requirements.

- *a convex set Θ and a matrix $\mathbf{M}$ embodying the data fidelity constraint and the domain (time or frequency) in which it is specified;*
- *a parameterized family of shrinkages $\{\mathcal{S}_\mu(\cdot)\}_\mu$, where the amount of shrinkage is controlled by μ: in the extreme cases $\mathcal{S}_0(\mathbf{Z}) = \mathbf{Z}$ and $\mathcal{S}_\infty(\mathbf{Z}) = \mathbf{0}$;*
- *a rule $F : \mu \mapsto F(\mu)$ to update the amount of shrinkage across iterations, and an initial $\mu^{(0)}$;*
- *an initial estimate $\mathbf{Z}^{(0)}$ of the seeked time-frequency representation;*
- *stopping parameters β and $\mathsf{i}_{\max}$.*

The proposed generic algorithm is described in Algorithm 1.

Algorithm 1 Generic Algorithm: $\mathcal{G}$

Require: $\Theta, \mathbf{M}, \{\mathcal{S}_\mu(\cdot)\}_\mu, \mu^{(0)}, F(\cdot), \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}$
  $\mathbf{U}^{(0)} = \mathbf{0}$;
  for $\mathsf{i} = 1$ to $\mathsf{i}_{\max}$ **do**
    $\mathbf{W}^{(\mathsf{i})} = \mathcal{P}_{\Theta,\mathbf{M}}(\mathbf{Z}^{(\mathsf{i}-1)} - \mathbf{U}^{(\mathsf{i}-1)})$
    $\mathbf{Z}^{(\mathsf{i})} = \mathcal{S}_{\mu^{(\mathsf{i}-1)}}\left(\mathbf{M}\mathbf{W}^{(\mathsf{i})} + \mathbf{U}^{(\mathsf{i}-1)}\right)$
    if $\frac{\|\mathbf{M}\mathbf{W}^{(\mathsf{i})} - \mathbf{Z}^{(\mathsf{i})}\|_\mathrm{F}}{\|\mathbf{M}\mathbf{W}^{(\mathsf{i})}\|_\mathrm{F}} \leq \beta$ **then**
      terminate
    else
      $\mathbf{U}^{(\mathsf{i})} = \mathbf{U}^{(\mathsf{i}-1)} + \mathbf{M}\mathbf{W}^{(\mathsf{i})} - \mathbf{Z}^{(\mathsf{i})}$.
      $\mu^{(\mathsf{i})} = F(\mu^{(\mathsf{i}-1)})$
    end if
  end for
  return $\mathbf{W}^{(\mathsf{i})}$ [and optionally $\mu^{(\mathsf{i})}$, $\mathbf{Z}^{(\mathsf{i})}$]

The notation $\mathbf{Z}^{(\mathsf{i})}$ highlights that the corresponding variable is in any use-case a sparse/structured time-frequency representation. The variable $\mathbf{U}^{(\mathsf{i})}$ is an intermediate time-frequency "residual" variable typical of ADMM / Douglas-Rachford. At iteration i, an estimate of $\mathbf{Z}_\mathsf{n}$ is $\hat{\mathbf{Z}}^{(\mathsf{i})} := \mathbf{Z}^{(\mathsf{i}-1)} - \mathbf{U}^{(\mathsf{i}-1)}$. The interpretation of the other variables is use-case dependent:

- **analysis flavor:** $\mathbf{M} := \mathbf{A}$ is the frequency analysis operator; $\mathbf{W}^{(\mathsf{i})}$ is an estimate of the time frames $\mathbf{X}_\mathsf{n}$, that satisfies the time-domain data-fidelity constraint Θ while being closest to $\hat{\mathbf{Z}}^{(\mathsf{i})}$ *in the time-frequency domain*; the algorithm outputs a time-domain estimate.
- **synthesis flavor:** $\mathbf{M} := \mathbf{I}$; $\mathbf{W}^{(\mathsf{i})}$ is a time-frequency estimate of $\mathbf{Z}_\mathsf{n}$; the data-fidelity constraint Θ is *expressed in the time-frequency domain*; the algorithm outputs a time-frequency estimate, from which it is possible to get a time-domain estimate by synthesis $\hat{\mathbf{X}}_\mathsf{n} := \mathbf{D}\mathbf{W}^{(\mathsf{i})}$.

Due to the expression of Θ respectively in the time domain and the time-frequency domain, the analysis and synthesis flavors will have different computational properties (for declipping mainly) as will be further discussed in Sections III-C and IV-C. Algorithm 1 is summarized as a generalized procedure $\mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu\}_\mu, \mu^{(0)}, F, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max})$.

C. Shrinkages for (social) sparsity

As noted in the requirements, we need to choose the family $\{\mathcal{S}_\mu(\cdot)\}_\mu$ of shrinkage operators.

For plain sparsity, either analysis or synthesis, we use the hard-thresholding operator $\mathcal{H}_\mathsf{k}(\mathbf{Z})$ that sets all but the k coefficients of largest magnitude in $\mathbf{Z}$ to zero (see e.g. [12]). In the case of analysis sparse modeling with $\mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}$ a forward frequency analysis operator (resp. $\mathbf{D} \in \mathbb{C}^{\mathsf{L}\times\mathsf{S}}$ a dictionary) we set $\mathcal{S}_\mu := \mathcal{H}_{\mathsf{P}-\mu}$ (resp. $\mathcal{S}_\mu := \mathcal{H}_{\mathsf{S}-\mu}$), for $\mu \in \mathbb{N}^+$, $0 \leq \mu \leq P$ (resp. $0 \leq \mu \leq S$).

For social sparsity (again, either analysis or synthesis), we choose the Persistent Empirical Wiener (PEW) operator [11] successfully used in [13] for audio declipping. This shrinkage promotes specific local time-frequency structures around each time-frequency point. Its specification explicitly requires choosing a time-frequency pattern described as a matrix $\Gamma \in \mathbb{R}^{(2\mathsf{F}+1)\times(2\mathsf{T}+1)}$ with binary entries.

Rows of Γ account for the frequency dimension and columns for the time dimension, in *local time-frequency coordinates*. Let $\mathbf{Z}_\mathsf{n} \in \mathbb{C}^{\mathsf{L}\times(2\mathsf{b}+1)}$ be a time-frequency representation. For clarity of presentation, we will now omit the n index and simply denote $\mathbf{Z}$ (and similarly for $\mathbf{X}$, $\mathbf{Y}$ later on when needed). As illustrated on Figure 2, consider ij the coordinates of a time-frequency point in $\mathbf{Z}$ and $\mathbf{P}_\mathsf{ij} := [\mathsf{i}-F, \mathsf{i}+F] \times [\mathsf{j}-T, \mathsf{j}+T]$ the indices corresponding to a time-frequency patch of size $(2\mathsf{F}+1)\times(2\mathsf{T}+1)$ centered in ij. The matrix $\mathbf{Z}_{\mathbf{P}_\mathsf{ij}} \in \mathbb{C}^{(2\mathsf{F}+1)\times(2\mathsf{T}+1)}$ is extracted from $\mathbf{Z}$ on these indices, with mirror-padding on the borders if needed.

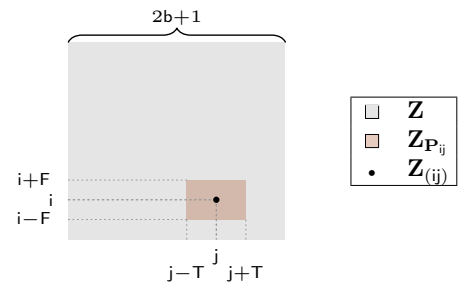


Fig. 2. Schematic representation of patch extraction from matrix $\mathbf{Z}$

Now that we have expressed how $\mathbf{Z}, \mathbf{Z}_{\mathbf{P}_\mathsf{ij}}$ and indexes are organized, we can define PEW using $\circ$ to denote the Hadamard product:

$$\mathcal{S}_\mu^{\text{PEW}}(\mathbf{Z}|\Gamma)_{(\mathsf{ij})} := \mathbf{Z}_{(\mathsf{ij})} \cdot \left(1 - \frac{\mu^2}{\|\mathbf{Z}_{\mathbf{P}_{\mathsf{ij}}} \circ \Gamma\|_2^2}\right)_+ . \quad (4)$$

Since $\|\mathbf{Z}_{\mathbf{P}_{\mathsf{ij}}} \circ \Gamma\|_2^2$ is the energy of $\mathbf{Z}$ restricted to a time-frequency neighborhood of ij of shape specified by Γ, the left hand side is zero as soon as this energy falls below μ^2. As such, PEW shrinkage effectively promotes structured sparsity.

Examples of patterns for music are given in Figure 3 and for speech in Figure 4. They are similar but at different time scales, given the different scales of stationarity in speech and music. The structures embedded in these patterns have various properties: Γ_1, with a frequency localized and time-spread support, will emphasize tonal content; vice-versa, Γ_2 will emphasize transients and attacks; Γ_3 is designed [14] to avoid pre-echo artifacts; patterns Γ_4 and Γ_5 are introduced to stress tonal transitions; finally, Γ_6 serves as a default pattern when no particular structure is identified.

Remarks: Here the subscript index k for each time-frequency pattern $\Gamma_\mathsf{k}, \mathsf{k} \in \{1..6\}$ is not a time frame index but counts the patterns within the collection.
On Fig. 3 the total time span for each Γ is 320 ms. On Fig. 4 the total time span reduces to 96 ms.

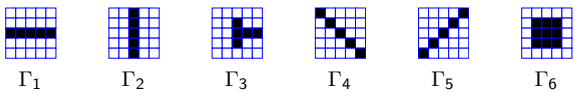


Fig. 3. Extended set of time-frequency neighborhoods used for music

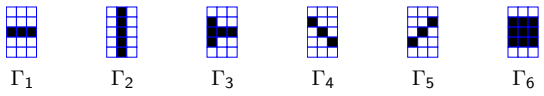


Fig. 4. Extended set of time-frequency neighborhoods used for music

Now that we described all the useful modeling and algorithmic tools used in our audio reconstruction scheme, sections III and IV will instantiate the enhancement procedure respectively for denoising and declipping and present experimental results.

III. Audio Denoising Use Case

Denoising is one of the most intensively studied inverse problems in audio signal processing. Whether it originates from the environment or the microphones, noise is an inevitable (and, usually, undesirable) component of audio recordings, calling for a denoising block in signal processing pipelines for applications such as speech recognition, sound classification, and many others.

A specific degraded model in case of additive noise writes:

$$\mathbf{Y}_\mathsf{n} = \mathbf{X}_\mathsf{n} + \mathbf{E}_\mathsf{n}, \quad (5)$$

with $\mathbf{E}_\mathsf{n}$ often modeled as white Gaussian noise of fixed variance σ^2 in the frame(s) around frame n.

A. Generalized projections for the denoising problem

A natural expression of the data-fidelity constraint is of the form $\|\hat{\mathbf{X}} - \mathbf{Y}\|_\mathrm{F} \leq \varepsilon$ for some ε. Heuristics to choose ε given an estimated variance σ^2 will be discussed in Section III-C.

In the analysis setting, with $\mathbf{M} := \mathbf{A}$, the data-fidelity constraint yields $\Theta := \{\mathbf{W} \mid \|\mathbf{W}-\mathbf{Y}\|_\mathrm{F} \leq \varepsilon\}$. In the synthesis setting, with $\mathbf{M} := \mathbf{I}$, we set $\Theta := \{\mathbf{W} \mid \|\mathbf{D}\mathbf{W} - \mathbf{Y}\|_\mathrm{F} \leq \varepsilon\}$. These choices hold both for plain and social versions.

In the analysis setting, assuming $\mathbf{A}^\mathsf{H}\mathbf{A} = \mathbf{I}$, the desired projection can be expressed in closed-form as:

$$\mathcal{P}_{\Theta,\mathbf{M}}(\mathbf{Z}) = \mathbf{A}^\mathsf{H}\mathbf{Z} - \left(\frac{\|\mathbf{A}^\mathsf{H}\mathbf{Z}-\mathbf{Y}\|_\mathrm{F}-\varepsilon}{\|\mathbf{A}^\mathsf{H}\mathbf{Z}-\mathbf{Y}\|_\mathrm{F}}\right)_+ \cdot \left(\mathbf{A}^\mathsf{H}\mathbf{Z} - \mathbf{Y}\right) \quad (6)$$

as shown in Appendix A for the more general case $\mathbf{A}^\mathsf{H}\mathbf{A} \propto \mathbf{I}$.

Hence, in this case, the cost of computing the generalized projection is dominated by matrix-vector products with $\mathbf{M}^\mathsf{H}$. When this can be done with a fast transform, the analysis flavor has low complexity. When complex transforms are used, to ensure the estimate is real, we replace $\mathbf{M}^\mathsf{H}\mathbf{Z}$ in (6) with $\Re(\mathbf{M}^\mathsf{H}\mathbf{Z}) - \Im(\mathbf{M}^\mathsf{H}\mathbf{Z})$ where $\Re(\cdot)$ and $\Im(\cdot)$ respectively denote the real and imaginary part.

For the synthesis version, assuming $\mathbf{D}\mathbf{D}^\mathsf{H} = \mathbf{I}$, the generalized projection again reduces algebraically to the closed-form expression:

$$\mathcal{P}_{\Theta,\mathbf{M}}(\mathbf{Z}) = \mathbf{Z} - \left(\frac{\|\mathbf{D}\mathbf{Z}-\mathbf{Y}\|_\mathrm{F}-\varepsilon}{\|\mathbf{D}\mathbf{Z}-\mathbf{Y}\|_\mathrm{F}}\right)_+ \cdot \mathbf{D}^\mathsf{H}(\mathbf{D}\mathbf{Z} - \mathbf{Y}). \quad (7)$$

B. Algorithms for the denoising inverse problem

We are now ready to instantiate the general algorithm $\mathcal{G}$ in the different cases.

1) Plain sparse audio denoisers: For both the analysis and the synthesis version, we instantiate the general algorithm $\mathcal{G}$ with the choices summarized in Table I.

TABLE I
Parameters of Algorithm 1 for the Plain Sparse Denoiser

Analysis	Synthesis
$\Theta = \{\mathbf{W} \mid \|\mathbf{W} - \mathbf{Y}\|_2 \leq \varepsilon\}$	$\Theta = \{\mathbf{W} \mid \|\mathbf{D}\mathbf{W} - \mathbf{Y}\|_2 \leq \varepsilon\}$
$\mathbf{M} = \mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \geq \mathsf{L}$	$\mathbf{M} = \mathbf{I} \in \mathbb{C}^{\mathsf{L}\times\mathsf{L}}$,
$\mathcal{S}_\mu(\cdot) = \mathcal{H}_{\mathsf{P}-\mu}(\cdot)$	$\mathcal{S}_\mu(\cdot) = \mathcal{H}_{\mathsf{S}-\mu}(\cdot)$,
$\mu^{(0)} = \mathsf{P} - 1$	$\mu^{(0)} = \mathsf{S} - 1$
$F : \mu \mapsto \mu - 1$	$F : \mu \mapsto \mu - 1$
$\mathbf{Z}^{(0)} = \mathbf{A}\mathbf{Y}$	$\mathbf{Z}^{(0)} = \mathbf{D}^\mathsf{H}\mathbf{Y}$

The choice of function F and initialization $\mu^{(0)}$ means that we start with a small number $\mathsf{P} - \mu^{(0)} = 1$ (resp. $\mathsf{S} - \mu^{(0)} = 1$) of nonzero coefficients for the sparse constraint which we relax gradually as iterations progress.

The practical choice of the stopping parameter β is driven by a compromise between quality and computation time and we will specify the values used in the experimental section.

Algorithm 1 with these parameters yields:

$$\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu(\cdot)\}_\mu, \mu^{(0)}, F, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}).$$

For the analysis version $\hat{\mathbf{X}} := \hat{\mathbf{W}}$, while for the synthesis version $\hat{\mathbf{X}} := \mathbf{D}\hat{\mathbf{W}}$.

2) Social sparse audio denoisers: For the social sparse versions of the denoising method, we change the sparsifying operator from $\mathcal{H}_{\mathsf{P}-\mu}(\cdot)$ to $\mathcal{S}^{\text{PEW}}_{\mu}(\cdot|\Gamma)$, as well as the update rule which becomes $F_\alpha : \mu \mapsto \alpha\mu$. The initial value $\mu^{(0)}$ may depend on the pattern Γ and will be specified in Section III-C. The resulting parameters are summarized in Table II.

TABLE II
PARAMETERS OF ALGORITHM 1 FOR THE SOCIAL SPARSE DENOISER

Analysis	Synthesis
$\Theta = \{\mathbf{W} \mid \|\mathbf{W}-\mathbf{Y}\|_2 \le \varepsilon\}$	$\Theta = \{\mathbf{W} \mid \|\mathbf{D}\mathbf{W}-\mathbf{Y}\|_2 \le \varepsilon\}$
$\mathbf{M} = \mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \ge \mathsf{L}$	$\mathbf{M} = \mathbf{I} \in \mathbb{C}^{\mathsf{L}\times\mathsf{L}}$,
$\mathcal{S}_\mu(\cdot) = \mathcal{S}^{\text{PEW}}_{\mu}(\cdot\|\Gamma)$	$\mathcal{S}_\mu(\cdot) = \mathcal{S}^{\text{PEW}}_{\mu}(\cdot\|\Gamma)$,
$\mu^{(0)}$: see Section III-C	$\mu^{(0)}$: see Section III-C
$F = F_\alpha : \mu \mapsto \alpha\mu$	$F = F_\alpha : \mu \mapsto \alpha\mu$
$\mathbf{Z}^{(0)} = \mathbf{A}\mathbf{Y}$	$\mathbf{Z}^{(0)} = \mathbf{D}^{\mathsf{H}}\mathbf{Y}$

A first version of the denoiser works with a *predefined* time-frequency pattern Γ and is compactly written as:

$$\begin{bmatrix}\hat{\mathbf{W}}(\Gamma)\\ \mu(\Gamma)\\ \mathbf{Z}(\Gamma)\end{bmatrix} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}^{\text{PEW}}_{\mu}(\cdot|\Gamma)\}_\mu, \mu^{(0)}, F_\alpha, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}).$$

A more adaptive denoiser uses this first version as a building brick to *select* the pattern Γ within a prescribed collection. Indeed, in order to get a fully adaptive denoising procedure, we design a method to automatically select the optimal Γ for the signal frames at stake. We call this step the "initialization loop". It consists in evaluating $\hat{\mathbf{W}}(\Gamma)$ with a small number of iterations (e.g. $\mathsf{i}^{\texttt{small}}_{\max} = 10$) for different patterns Γ.

Given a predefined set of time-frequency patterns $\{\Gamma_\mathsf{k}\}_{\mathsf{k}=1}^{\mathsf{K}}$ and initial threshold values $\mu^{(0)}_\mathsf{k}$ that will be specified in Section III-C, one can compute $\hat{\mathbf{W}}_\mathsf{k} := \hat{\mathbf{W}}(\Gamma_\mathsf{k})$ for $1 \le \mathsf{k} \le \mathsf{K}$, and similarly $\mu_\mathsf{k} := \mu(\Gamma_\mathsf{k})$ and $\mathbf{Z}_\mathsf{k} := \mathbf{Z}(\Gamma_\mathsf{k})$. Then, the idea is that the best estimate $\hat{\mathbf{W}}_\mathsf{k}$ should produce a residual with spectrum closest to that of Additive White Gaussian Noise (AWGN), which is by definition flat. Thus, we select the pattern $\Gamma_{\mathsf{k}^\star}$ yielding a residual with time-frequency representation of highest entropy.

For a given k, we can define the resulting time-frequency residual: $\mathbf{R}_\mathsf{k} := \mathbf{M}\hat{\mathbf{W}}_\mathsf{k} - \mathbf{Z}^{(0)}$. Computing a Q-bin histogram of the modulus of its entries yields $\hat{\mathrm{p}}$, an empirical probability distribution, which (empirical) entropy is

$$\mathrm{e}_\mathsf{k} = -\sum_{\mathsf{q}=1}^{\mathsf{Q}} \hat{\mathrm{p}}_\mathsf{q} \log_2(\hat{\mathrm{p}}_\mathsf{q}). \tag{8}$$

A heuristic to choose Q is the Herbert-Sturges rule [15]

$$\mathsf{Q} = \lfloor 1 + \log_2(\#\mathbf{R}_\mathsf{k}) \rfloor, \tag{9}$$

where $\lfloor\cdot\rfloor$ is the floor function and $\#\mathbf{R}_\mathsf{k} = \mathsf{L} \times (2\mathsf{b}+1)$ is the number of entries in the matrix $\#\mathbf{R}_\mathsf{k}$. The values considered in the experiments of Section III-C lead to $\mathsf{Q} \in \{13, 15\}$.

Once the best pattern $\Gamma_{\mathsf{k}^\star}$ is chosen as just described, we run Algorithm 1 with the parameters of Table II and warm-started $\mu^{(0)}$ and $\mathbf{Z}^{(0)}$, with a sufficiently large $\mathsf{i}_{\max}$ (typically $\mathsf{i}^{\texttt{large}}_{\max} = 10^6$) to get

$$\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}^{\text{PEW}}_{\mu}(\cdot|\Gamma_{\mathsf{k}^\star})\}_\mu, \mu_{\mathsf{k}^\star}, F_\alpha, \mathbf{Z}_{\mathsf{k}^\star}, \beta, \mathsf{i}^{\texttt{large}}_{\max}).$$

The pseudo-code of the adaptive social denoiser for a given block of adjacent frames $\mathbf{Y} \in \mathbb{R}^{\mathsf{L}\times(2\mathsf{b}+1)}$ is given in Algorithm 2. Again, for the analysis version $\hat{\mathbf{X}} := \hat{\mathbf{W}}$, while for the synthesis version $\hat{\mathbf{X}} := \mathbf{D}\hat{\mathbf{W}}$.

Algorithm 2 Adaptive Social Sparse Denoisers

Require: $\mathbf{Y}$, ε, $\mathbf{A}$ or $\mathbf{D}$, $\{\Gamma_\mathsf{k}\}_\mathsf{k}$, $\{\mu^{(0)}_\mathsf{k}\}_\mathsf{k}$, α, β, $\mathsf{i}^{\texttt{small}}_{\max}$, $\mathsf{i}^{\texttt{large}}_{\max}$
  set parameters from Table II
  for each k **do**

$$\begin{bmatrix}\hat{\mathbf{W}}_\mathsf{k}\\ \mu_\mathsf{k}\\ \mathbf{Z}_\mathsf{k}\end{bmatrix} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}^{\text{PEW}}_{\mu}(\cdot|\Gamma_\mathsf{k})\}_\mu, \mu^{(0)}_\mathsf{k}, F_\alpha, \mathbf{Z}^{(0)}, \beta, \mathsf{i}^{\texttt{small}}_{\max})$$

    Compute e_k as in (8)
  end for
  $\mathsf{k}^\star := \arg\max_\mathsf{k} \mathrm{e}_\mathsf{k}$
  $\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}^{\text{PEW}}_{\mu}(\cdot|\Gamma_{\mathsf{k}^\star})\}_\mu, \mu_{\mathsf{k}^\star}, F_\alpha, \mathbf{Z}_{\mathsf{k}^\star}, \beta, \mathsf{i}^{\texttt{large}}_{\max})$.
  return $\hat{\mathbf{W}}$

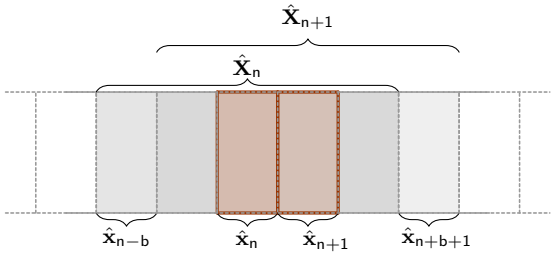


Fig. 5. Segment processing for frame n and frame $\mathsf{n}+1$.

3) Post-processing and overlap-add synthesis: We recall that the denoiser is applied in an frame-based scenario: given noisy frame(s) $\mathbf{Y}_\mathsf{n}$, the denoisers output estimated frame(s) $\hat{\mathbf{X}}_\mathsf{n}$ which need to be transformed back into a full time-domain signal $\mathbf{x}$. For this, we first need to extract from $\hat{\mathbf{X}}_\mathsf{n}$ a single estimated frame $\hat{\mathbf{x}}_\mathsf{n}$:

- in the plain sparse case, this is straightforward as $\hat{\mathbf{X}}_\mathsf{n} \in \mathbb{R}^{\mathsf{L}\times 1}$ is already a vector;
- for the social sparse case, we set $\hat{\mathbf{x}}_\mathsf{n} = \hat{\mathbf{X}}_\mathsf{n}(:, \mathsf{b}+1)$ to be the central column of the matrix $\hat{\mathbf{X}}_\mathsf{n}$, see Fig. 5.

Given the estimated frames $\{\hat{\mathbf{x}}_\mathsf{n}\}_\mathsf{n}$, and before the final overlap-add that will lead to the full time-domain estimate $\hat{\mathbf{x}}$, we perform a simple frequency-domain Wiener filtering on each $\hat{\mathbf{x}}_\mathsf{n}$ similar to the one used in the Block-Thresholding algorithm [5] which we will use as a comparison in Section III-C. Such a Wiener filtering requires an estimation of the noise power σ^2, as well as an estimation of the signal power, both in the frequency domain. For the latter, we use the squared magnitudes of $\mathbf{A}\hat{\mathbf{x}}$ (resp. of $\mathbf{D}^{\mathsf{H}}\hat{\mathbf{x}}$). Oracle values of σ^2 will be used in the experiments. Practically, we observed that this post-processing is useful at very low SNR (*i.e* 0 dB) where we observe "musical noise" effect.

Finally, overlap-add synthesis is performed, taking into account the windows that were applied onto the frames to get the noisy frames $\mathbf{Y}_\mathsf{n}$.

C. Experimental Study

This section aims at comparing effects of the different shrinkages (plain or social), the different models (synthesis or analysis), and the degradation level on the audio denoising performance.

a) Datasets: We conduct experiments on excerpts from the RWC Music Database [16]. We use the "Pop" and "Jazz" genres as is, and subcategorize the "Classic" genre (Vocals, Chamber, Symphonies), leading to 5 subsets. All the tracks are sufficiently diverse to reflect the robustness of the approach on different audio content. For each subset of the database, we contaminated an excerpt of each available track in order to get around one hour of noisy material for each category. We also perform experiments on the TIMIT database [17] for evaluation on speech content. All the audio examples used here are also down-sampled to 16 kHz.

b) Performance measures: We use as a first objective recovery performance numerical measure the Signal-to-Noise Ratio (SNR) difference between the noisy and enhanced signals. We also compare the perceptual speech quality with the objective MOS-PESQ scores [18], and analyse the intelligibility through the Short-Time Objective Intelligibility index (STOI) [19]. For music, we compare the perceptual quality on the entire RWC excerpts collection with the Objective Difference Grade (ODG) PEAQ scores [20]. Finally, for all methods we compare computation times in seconds.

c) Compared methods: We consider the plain sparse, plain cosparse, social sparse and social cosparse denoisers, as well as the state-of-the-art time-frequency block thresholding (BT) [5]. The main parameters are set as follows:

- frame size: $\mathsf{L} = 64$ ms for music $\mathsf{L} = 32$ ms for speech;
- overlap, 75%;
- number of overlapping segments for social denoisers: $\mathsf{b} = 5$ for music, $\mathsf{b} = 1$ for speech;
- time-frequency patterns for social denoisers: $\{\Gamma_\mathsf{k}\}_{\mathsf{k}=1}^{\mathsf{K}}$ presented on Fig. 3 for music and Fig. 4 for speech.
- stopping criteria $\beta = 10^{-3}$, $\mathrm{i}_{\max}^{\mathtt{small}} = 10$, $\mathrm{i}_{\max}^{\mathtt{large}} = 10^6$;

The time-frequency synthesis/analysis operators are:

- **Synthesis operator**: $\mathbf{D}$ is the inverse DFT of redundancy R, that is to say $\mathbf{D} \in \mathbb{C}^{\mathsf{L}\times\mathsf{S}}$ with $\mathsf{S} = (\mathsf{R}\times\mathsf{L})$ and $\mathbf{D}_{\mathsf{ls}} := \mathsf{S}^{-1/2} e^{j\frac{2\pi \mathsf{ls}}{\mathsf{S}}}$. One can check that $\mathbf{D}\mathbf{D}^{\mathsf{H}} = \mathbf{I}$;
- **Analysis operator**: $\mathbf{A}$ is the forward DFT of redundancy R, that is to say $\mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}$ with $\mathsf{P} = (\mathsf{R}\times\mathsf{L})$ and $\mathbf{A}_{\mathsf{pl}} = \mathsf{P}^{-1/2} e^{-j\frac{2\pi \mathsf{pl}}{\mathsf{P}}}$. Again, one can check that $\mathbf{A}^{\mathsf{H}}\mathbf{A} = \mathbf{I}$.

Practically, products with $\mathbf{A}$ (resp. $\mathbf{D}^{\mathsf{H}}$), are done using the FFT of size P (resp. S) on a zero-padded signal of initial length L. Similarly, products with $\mathbf{A}^{\mathsf{H}}$ (resp. $\mathbf{D}$), are done by truncating the inverse fast transform.

All denoisers require a parameter ε ruling the l_2 regularization for the denoising constraint. For the plain sparse denoisers, ε is set to $\sigma\sqrt{\sum_{\mathsf{j}=1}^{\mathsf{L}} \mathbf{w}_\mathsf{j}}$, with $\mathbf{w}_\mathsf{j}$ the j^{th} entry of the window $\mathbf{w}$ and σ^2 the known noise variance. For the adaptive social sparse denoisers, we scale ε to $(2\mathsf{b}+1)\sigma\sqrt{\sum_{\mathsf{j}=1}^{\mathsf{L}} \mathbf{w}_\mathsf{j}}$.

The adaptive social sparse denoisers also require to set $\mu_\mathsf{k}^{(0)}$ and α (see Algorithm 2). To adapt these parameters to the local peak audio level $\|\mathrm{vec}(\mathbf{Y})\|_\infty$ and to the number of active bins in the time-frequency pattern Γ_k, we set

$$\mu_\mathsf{k}^{(0)} := \|\Gamma_\mathsf{k}\|_0 \times \|\mathrm{vec}(\mathbf{Y})\|_\infty \quad (10)$$

$$\alpha := \min\left(\frac{\sigma}{\sqrt{\mathrm{var}(\mathrm{vec}(\mathbf{Y}))}}, 0.99\right), \quad (11)$$

where $\mathrm{vec}(\cdot)$ vectorizes the matrix. This parameterization reflects the "instantaneous" SNR in the region being processed. The two parameters α and μ rule how aggressively the sparse regularization is performed.

d) Pilot study: Given the large combinatorics of experiments related to all possible configurations (plain/social, analysis/synthesis, redundancy factor) and noise levels, we performed a first pilot study for two input noise levels (5 dB, 20 dB). Each configuration was tested over five 10 second excerpts from the RWC database covering the five music genre subsets. The average SNR improvements, as well the average computation times[2] (relative to the audio duration) are summarized in Table III.

TABLE III
PILOT STUDY: SNR IMPROVEMENTS (ΔSNR) AND PROCESSING TIMES RELATIVE TO AUDIO DURATION ($\times$RT) FOR ALL CONFIGURATIONS

	Plain				Social			
	Analysis		Synthesis		Analysis		Synthesis	
	ΔSNR	$\times$RT	ΔSNR	$\times$RT	ΔSNR	$\times$RT	ΔSNR	$\times$RT
$\mathsf{R}=1$	7.04	2.9	7.04	4.0	7.70	5.9	7.70	8.1
$\mathsf{R}=2$	7.29	10.0	7.30	13.3	7.75	12.8	7.73	16.6
$\mathsf{R}=4$	7.26	20.9	7.26	26.3	7.78	24.5	7.74	31.5

(a) Input SNR: 5 dB

	Plain				Social			
	Analysis		Synthesis		Analysis		Synthesis	
	ΔSNR	$\times$RT	ΔSNR	$\times$RT	ΔSNR	$\times$RT	ΔSNR	$\times$RT
$\mathsf{R}=1$	3.17	5.9	3.17	8.0	3.21	2.2	3.21	2.9
$\mathsf{R}=2$	3.32	13.9	3.31	17.4	3.18	5.0	3.19	6.5
$\mathsf{R}=4$	3.29	26.4	3.29	34.1	3.24	10.9	3.23	13.8

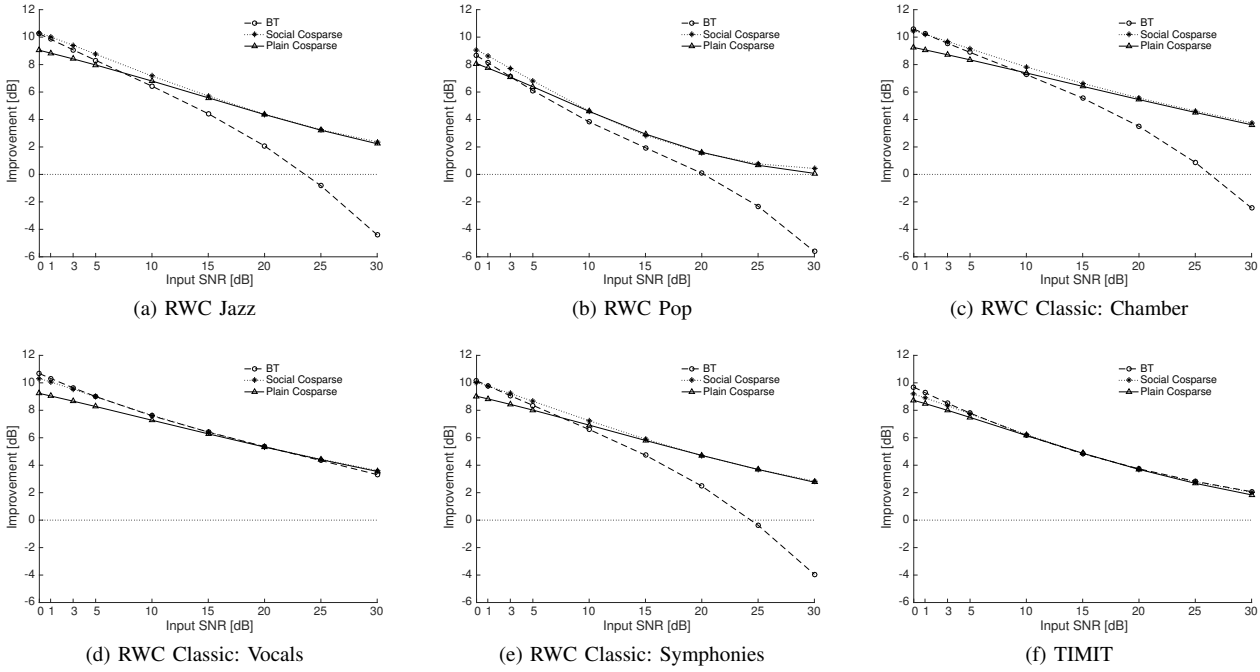
(b) Input SNR: 20 dB

We observe that:

- For each noise level, each redundancy, and each thresholding operator, the performance of the analysis and synthesis models in decibels is almost identical, while the analysis version is by 20% to nearly 40% faster that the synthesis version. As a result the rest of the experiments are conducted only with the analysis version.
- All other factors being equal, the computation time is roughly proportional to the redundancy R, while the improvement in ouput SNR is often very limited. In the rest of the experiments we thus choose $\mathsf{R} = 2$, which seems to give the best compromise (and in fact, even the best performance in many configurations). It also enables a transparent comparison with the baseline defined by block thresholding.

e) Denoising performance: Given the pilot study, we now focus on the cosparse denoisers (plain and social) as well

[2]All reported computation times were measured using a Matlab® implementation of the algorithms on a laptop equipped with a 2.8 Ghz Intel® Core™ i7 processor and 16 GB of RAM memory.



(a) RWC Jazz (b) RWC Pop (c) RWC Classic: Chamber

(d) RWC Classic: Vocals (e) RWC Classic: Symphonies (f) TIMIT

Fig. 6. Denoising Numerical Results: SNR improvement [dB]

as block thresholding, all with redundancy $\mathsf{R} = 2$. We consider nine input SNR levels in dB: $\{0, 1, 3, 5, 10, 15, 20, 25, 30\}$ and work with the full datasets.

Figure 6 shows averaged SNR improvements over each of the 5 music subsets as well as the TIMIT speech dataset.

Results on Fig. 6 show that either the social cosparse or the plain cosparse algorithms outperform Block Thresholding (BT) on the denoising task for almost every category of audio content. The benefit given by the social flavour of the algorithm is widely seen at low SNRs where the social method and BT have comparable performance. Indeed, the social version perform 1 to 3 dB better than the simple cosparse version for input SNRs below 10. These results strengthen the idea that adaptiveness can be beneficial for highly degraded conditions. Moreover, the simple sparse approach catch up with or even overstep the social one for higher SNRs. This way, we can guess that recovering hidden structure is optional as the signal is already well clustered in the time-frequency plane for light noise conditions. The difference between our approaches and BT increases with the input SNR. We note a significant contrast at high SNR where BT underperforms by more than 6 dB in the less favorable configuration. This might be because BT strongly relies on the noise model whereas cosparse and social cosparse methods try to emphasize the signal itself.

We gathered standard deviation informations associated to Figure 6 and results demonstrate that the plain cosparse denoiser produces less variable results as the standard deviation is the lowest for this technique in 80% of the tested cases. We also notice that, without considering any specific algorithm, the improvement variability seems to increase with the input SNR. Indeed, for light noise conditions, the standard deviation reaches up to 3.29 dB for BT on the RWC "Solo" musical excerpts.

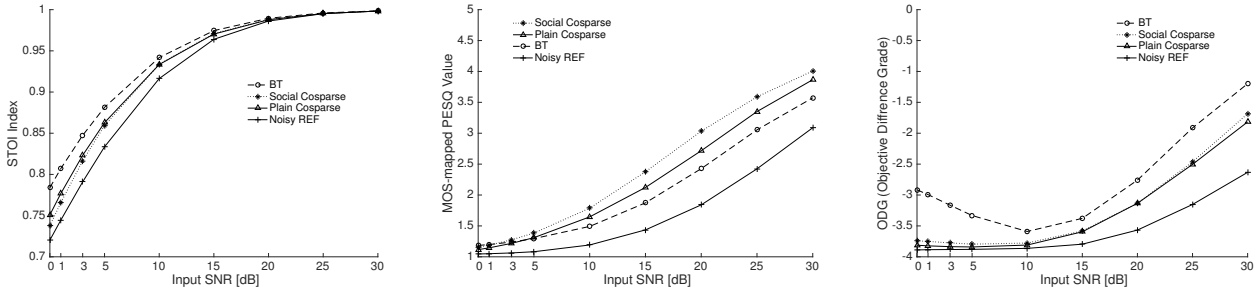
TABLE IV
COMPUTATIONAL PERFORMANCE OF COSPARSE DENOISERS

Input SNR [dB]	Plain cosparse		Social cosparse	
	Δ SNR	$\times$ RT	Δ SNR	$\times$ RT
0	9.45	7.8	9.50	16.6
5	7.62	10.3	7.76	12.0
10	5.91	12.6	6.03	9.2
15	4.33	16.6	4.50	7.0
20	3.02	22.1	3.18	5.5

Figure 7 shows averaged STOI/PESQ/PEAQ performance over each of the 5 music subsets as well as the TIMIT speech dataset. Even if Fig. 6d and 6f do not show clear superiority of one or another method on SNR improvement for voice based audio content, Fig. 7b reveals improved objective speech quality (PESQ metric) for both social and plain cosparse denoisers.

f) Computation time: For the social case, the computational cost is driven by the shrinkage (PEW) and the projection steps. However, evaluating PEW shrinkage is relatively fast, as it can be computed through 2-D convolution in the time-frequency domain. Besides, since we set low $\mathsf{i}_{\max}$ for the initialization loop, the choice of Γ is quite fast and adds only $(\mathsf{b}-1) \times \mathsf{i}_{\max}^{\mathtt{small}}$ iterations compared to the case where only one time-frequency pattern is considered. These properties allow to expect the social cosparse denoiser to have runtime comparable to that of the plain cosparse denoiser.

Table IV displays processing times in seconds for both



(a) TIMIT: STOI Index (b) TIMIT: MOS-Mapped PESQ Value (c) RWC: PEAQ ODG score

Fig. 7. Denoising Objective Quality and Intelligibility Results

denoising procedures. These computational comparisons are conducted on a single 10 second sound excerpt for each genre. The SNR improvements are in line with what was previously observed on larger datasets with very similar performance of both methods. However we note very different behaviors between the two cosparse denoisers in terms of runtime. While the plain flavor is fastest at low SNR, the social version is fastest at high SNR. This suggests that in practice the choice of one of these methods might be rather driven by speed considerations than by denoising performance.

IV. Audio Declipping Use Case

With denoising, declipping is another well-known problem in signal processing. Magnitude saturation can occur at different steps in the acquisition, reproduction or analog-to-digital conversion process. Restoring saturated signal is of great interest for many applications in digital communications, image processing or audio. In the latter, while light to moderate clipping cause only some audible clicks and pops, more severe saturation highly affect original signals which sound contaminated by rattle noise. The perceived degradation depends on the clipping level and the original signal. More recently, studies [21], [22] showed the negative impact of clipped signals when used in signal-processing pipelines leading to recognition, transcription or classification applications.

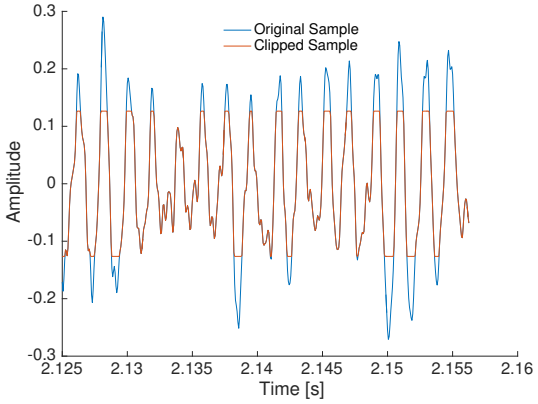
In the following section, we use the idealized hard-clipping model below. Although simple, it correctly approximates the magnitude saturation and allows to split up the samples into a clipped set and a reliable set:

$$\mathbf{y}_{ij} = \begin{cases} \mathbf{x}_{ij} & \text{for } |\mathbf{x}_{ij}| \leq \tau; \\ \operatorname{sgn}(\mathbf{x}_{ij})\tau & \text{otherwise;} \end{cases} \tag{12}$$

with $\mathbf{y}_{ij}$ (resp. $\mathbf{x}_{ij}$) a sample from $\mathbf{Y}_n$ (resp. $\mathbf{X}_n$) and τ the hard-clipping level. A visual example of such a degradation is given on figure 8. In real settings where softer saturation occurs, this model can be enforced with appropriate data pre-processing.

A. Generalized projections for the declipping problem

Denote Ω_+ (resp. Ω_-) the collection of indices ij of the samples in matrix $\mathbf{Y}$ affected by positive (resp. negative) magnitude clipping. Similarly denote Ω_r the indices of the reliable samples (not affected by clipping), and for any of these sets Ω define $\mathbf{V}_\Omega$ the matrix formed by keeping only the entries of $\mathbf{V}$ indexed by Ω and setting the rest to zero.



(a) Signal waveforms

Fig. 8. Clipped signal example

The data-fidelity constraint can now be expressed for the analysis setting with $\mathbf{M} := \mathbf{A}$ by

$$\Theta := \left\{ \mathbf{W} \mid \begin{array}{c} \mathbf{W}_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ \mathbf{W}_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ \mathbf{W}_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}$$

while for the synthesis setting, with $\mathbf{M} := \mathbf{I}$, we set

$$\Theta := \left\{ \mathbf{W} \mid \begin{array}{c} (\mathbf{DW})_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ (\mathbf{DW})_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ (\mathbf{DW})_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}.$$

Similarly to the denoising use-case, these choices hold for both plain and social versions.

In the analysis setting, the desired projection reduces to component wise magnitude constraints (see. Appendix B) and can be expressed as:

$$[\mathcal{P}_{\Theta,\mathbf{M}}(\mathbf{Z})]_{(ij)} = \begin{cases} \mathbf{Y}_{(ij)} & \text{if ij} \in \Omega_r; \\ (\mathbf{M}^H\mathbf{Z})_{(ij)} & \text{if } \begin{cases} \text{ij} \in \Omega_+, (\mathbf{M}^H\mathbf{Z})_{(ij)} \geq \tau; \\ \text{or} \\ \text{ij} \in \Omega_-, (\mathbf{M}^H\mathbf{Z})_{(ij)} \leq -\tau; \end{cases} \\ \operatorname{sgn}(\mathbf{Y}_{(ij)})\tau & \text{otherwise.} \end{cases}$$

In this case, matrix-vector products with $\mathbf{M}^H$ dominates the computing cost of the generalized projection. When this can

be done with a fast transform, the analysis flavor has once again low complexity.

For the synthesis case, the projection step is not that straightforward and needs to be computed with another nested iterative procedure. Even if exploiting the tight-frame property on $\mathbf{D}$ can help building an efficient algorithm for the projection, the overall computation cost for the synthesis flavor however remains substantially higher than the analysis version. To get more details on this projection, refer to Appendix B.

B. Algorithms for the declipping inverse problem

With all the steps defined, we can now instantiate the general algorithm $\mathcal{G}$ in the different cases.

1) Plain sparse audio declippers: Similarly to denoising, for both the analysis and the synthesis version, we instantiate the general algorithm $\mathcal{G}$ by choosing the operators described in Table V.

The update rule F for μ is set to gradually decrease μ by 1 at each iteration, starting from $\mu^{(0)} = \mathsf{P} - 1$ for the analysis case (resp. $\mu^{(0)} = \mathsf{S} - 1$ for the synthesis case). This way, we relax the sparse constraint the same way we do it for denoising.

TABLE V
PARAMETERS OF ALGORITHM 1 FOR THE PLAIN SPARSE DECLIPPER

Analysis	**Synthesis**
$\Theta = \left\{ \mathbf{W} \mid \begin{array}{c} \mathbf{W}_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ \mathbf{W}_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ \mathbf{W}_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}$	$\Theta = \left\{ \mathbf{W} \mid \begin{array}{c} (\mathbf{DW})_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ (\mathbf{DW})_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ (\mathbf{DW})_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}$
$\mathbf{M} = \mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \geq \mathsf{L}$	$\mathbf{M} = \mathbf{I} \in \mathbb{C}^{\mathsf{L}\times\mathsf{L}}$,
$\mathcal{S}_\mu(\cdot) = \mathcal{H}_{\mathsf{P}-\mu}(\cdot)$	$\mathcal{S}_\mu(\cdot) = \mathcal{H}_{\mathsf{S}-\mu}(\cdot)$,
$\mu^{(0)} = \mathsf{P} - 1$	$\mu^{(0)} = \mathsf{S} - 1$
$F : \mu \mapsto \mu - 1$	$F : \mu \mapsto \mu - 1$
$\mathbf{Z}^{(0)} = \mathbf{AY}$	$\mathbf{Z}^{(0)} = \mathbf{D}^\mathsf{H}\mathbf{Y}$

Iterating Algorithm 1 with the parameters described above gives a declipped estimate $\hat{\mathbf{W}}$ such that:

$$\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu(\cdot)\}_\mu, \mu^{(0)}, F, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}).$$

We recall that for the analysis version $\hat{\mathbf{X}} := \hat{\mathbf{W}}$, while for the synthesis version $\hat{\mathbf{X}} := \mathbf{D}\hat{\mathbf{W}}$.

2) Social sparse audio declippers: Similarly to the social sparse audio denoising procedure, we change the sparsifying operator to $\mathcal{S}_\mu^{\text{PEW}}(\cdot \mid \Gamma)$ and the update rule which we set now to $F_\alpha : \mu \mapsto \alpha\mu$. The initial value $\mu^{(0)}$ may also depend here on the pattern Γ and will be precised in Section IV-C.

The resulting parameters are summarized in Table VI.

The social declipper with a *predefined* time-frequency pattern Γ is compactly written as:

$$\begin{bmatrix} \hat{\mathbf{W}}(\Gamma) \\ \mu(\Gamma) \\ \mathbf{Z}(\Gamma) \end{bmatrix} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu^{\text{PEW}}(\cdot|\Gamma)\}_\mu, \mu^{(0)}, F_\alpha, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}),$$

The adaptive social declipper uses this to select the "optimal" pattern Γ within a prescribed collection $\{\Gamma_\mathsf{k}\}_{\mathsf{k}=1}^\mathsf{K}$ for the processed signal region, by running few iterations of the algorithm (typically $\mathsf{i}_{\max}^{\mathtt{small}} = 10$). The whiteness of the residual is evaluated with the same entropy criterion (8) as in denoising, which maximization yields the selected pattern $\Gamma_{\mathsf{k}^\star}$.

TABLE VI
PARAMETERS OF ALGORITHM 1 FOR THE SOCIAL SPARSE DECLIPPER

Analysis	**Synthesis**
$\Theta = \left\{ \mathbf{W} \mid \begin{array}{c} \mathbf{W}_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ \mathbf{W}_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ \mathbf{W}_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}$	$\Theta = \left\{ \mathbf{W} \mid \begin{array}{c} (\mathbf{DW})_{\Omega_r} = \mathbf{Y}_{\Omega_r}; \\ (\mathbf{DW})_{\Omega_+} \succcurlyeq \mathbf{Y}_{\Omega_+}; \\ (\mathbf{DW})_{\Omega_-} \preccurlyeq \mathbf{Y}_{\Omega_-}. \end{array} \right\}$
$\mathbf{M} = \mathbf{A} \in \mathbb{C}^{\mathsf{P}\times\mathsf{L}}, \mathsf{P} \geq \mathsf{L}$	$\mathbf{M} = \mathbf{I} \in \mathbb{C}^{\mathsf{L}\times\mathsf{L}}$,
$\mathcal{S}_\mu(\cdot) = \mathcal{S}_\mu^{\text{PEW}}(\cdot\|\Gamma)$	$\mathcal{S}_\mu(\cdot) = \mathcal{S}_\mu^{\text{PEW}}(\cdot\|\Gamma)$,
$\mu^{(0)}$: see Section IV-C	$\mu^{(0)}$: see Section IV-C
$F = F_\alpha : \mu \mapsto \alpha\mu$	$F = F_\alpha : \mu \mapsto \alpha\mu$
$\mathbf{Z}^{(0)} = \mathbf{AY}$	$\mathbf{Z}^{(0)} = \mathbf{D}^\mathsf{H}\mathbf{Y}$

Correspondingly, the first value $\mu_{(\mathsf{k})}^{(0)}$ and the update rule F_α as well as the time-frequency patterns $\{\Gamma_\mathsf{k}\}_{\mathsf{k}=1}^\mathsf{K}$ are essential for the algorithm to provide improvements. These will be specified in Section IV-C.

Once the best time-frequency pattern is selected, we run Algorithm 1 with the parameters listed in Table VI and a sufficiently large $\mathsf{i}_{\max}$ (typically $\mathsf{i}_{\max}^{\mathtt{large}} = 10^6$) to get

$$\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu^{\text{PEW}}(\cdot|\Gamma_{\mathsf{k}^\star})\}_\mu, \mu_{\mathsf{k}^\star}, F_\alpha, \mathbf{Z}_{\mathsf{k}^\star}, \beta, \mathsf{i}_{\max}^{\mathtt{large}}).$$

The pseudo-code of the adaptive social declipper for a given block of adjacent frames $\mathbf{Y} \in \mathbb{R}^{\mathsf{L}\times(2\mathsf{b}+1)}$ is given in Algorithm 3. Again, for the analysis version $\hat{\mathbf{X}} := \hat{\mathbf{W}}$, while for the synthesis version $\hat{\mathbf{X}} := \mathbf{D}\hat{\mathbf{W}}$.

Algorithm 3 Adaptive Social Sparse Declipper

Require: $\mathbf{Y}$, ε, $\mathbf{A}$ or $\mathbf{D}$, $\{\Gamma_\mathsf{k}\}_\mathsf{k}$, $\{\mu_\mathsf{k}^{(0)}\}_\mathsf{k}$, α, β, $\mathsf{i}_{\max}^{\mathtt{small}}$, $\mathsf{i}_{\max}^{\mathtt{large}}$
  set parameters from Table II, $\alpha = 1$
  for each k **do**

$$\begin{bmatrix} \hat{\mathbf{W}}_\mathsf{k} \\ \mu_\mathsf{k} \\ \mathbf{Z}_\mathsf{k} \end{bmatrix} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu^{\text{PEW}}(\cdot|\Gamma_\mathsf{k})\}_\mu, \mu_\mathsf{k}^{(0)}, F_\alpha, \mathbf{Z}^{(0)}, \beta, \mathsf{i}_{\max}^{\mathtt{small}})$$

    Compute e_k as in (8)
  end for
  $\mathsf{k}^\star := \arg\max_\mathsf{k} \mathsf{e}_\mathsf{k}$, $\alpha = 0.99$
  $\hat{\mathbf{W}} := \mathcal{G}(\Theta, \mathbf{M}, \{\mathcal{S}_\mu^{\text{PEW}}(\cdot|\Gamma_{\mathsf{k}^\star})\}_\mu, \mu_{\mathsf{k}^\star}, F_\alpha, \mathbf{Z}_{\mathsf{k}^\star}, \beta, \mathsf{i}_{\max}^{\mathtt{large}})$.
  return $\hat{\mathbf{W}}$

3) Overlap-add synthesis: As in denoising, the overall declipped signal is obtained by overlap-add, here without any Wiener filtering post-processing.

C. Experimental Study

This experimental validation aims at comparing the impact of the different modelings and saturation levels on the declipping performance for audio signals.

a) Datasets: We direct experiments on the same material (RWC and TIMIT databases) used in the denoising section (Section III-C). Each excerpt was first amplitude normalized ($\|\text{vec}(\mathbf{X})\|_\infty = 1$) then artificially clipped following the hard-clipping model in (12) for various values of $0 < \tau < 1$.

b) Performance measure: We use as a first enhancement measurement the Signal-to-Distortion Ratio (SDR) [23] difference (ΔSDR) between the clipped and the recovered signal. For speech, we also study the perceptual quality improvements through the MOS-PESQ scores and the objective intelligibility with the STOI index. Over all musical excerpts from the RWC subset, we evaluate the perceptual audio quality with the ODG PEAQ score. Results produced on Figures 9 and 10 display averaged measurements over all available samples.

c) Compared methods: Similarly to the denoising section, we consider the plain sparse, plain cosparse, social sparse and social cosparse declippers. We set the common parameters for the algorithms as listed below.

- Frame size $\mathsf{L} = 64$ ms for music $\mathsf{L} = 32$ ms for speech;
- Overlap, 75%;
- Accurracy $\beta = 10^{-3}, \mathsf{i}_{\max}^{\mathtt{small}} = 10, \mathsf{i}_{\max}^{\mathtt{large}} = 10^6$;
- Analysis operator, $\mathbf{A} = \mathrm{DFT}$;
- Synthesis operator, $\mathbf{D} = \text{inverse DFT}$.

Considering the adaptive social sparse declipper and similarly to denoising, we set the collection of time-frequency patterns $\{\Gamma_\mathsf{k}\}_{\mathsf{k}=1}^{\mathsf{K}}$ to match the one presented on Fig. 3 for music and Fig. 4 for speech. The specific choice of $\mu_\mathsf{k}^{(0)} := \|\Gamma\|_0 \times (1-\tau)$ is motivated by the sparsity degree of the time-frequency neighborhood considered. With this parameterization, the regularization behavior is initialized inversely proportional to the maximum magnitude of the clipped signal, allowing highly clipped configurations to retain sparser regularization. Contrarily to the social sparse denoising method, we notice better improvements when the μ parameter is *not* updated during the initialization loop (*i.e.* $\alpha = 1$). Once the proper $\Gamma_{\mathsf{k}^\star}$ is selected, so far, we obtained the better declipping results with μ following a geometric progression of common ratio α with $\alpha = 0.99$. We finally set the number of overlapping segments to $\mathsf{b} = 5$ for music, $\mathsf{b} = 1$ for speech (*i.e.* 11 frames or 3 frames at a time).

TABLE VII
PROCESSING TIMES COMPARISON (PLAIN SPARSE DECLIPPERS)

	Input SDR: 5 dB				Input SDR: 20 dB			
	Analysis		Synthesis		Analysis		Synthesis	
	ΔSDR	$\times$RT	ΔSDR	$\times$RT	ΔSDR	$\times$RT	ΔSDR	$\times$RT
$\mathsf{R}=1$	8.90	10.7	8.90	1183.8	10.37	7.5	10.37	1041.6
$\mathsf{R}=2$	9.73	41.7	9.81	2353.2	11.04	24.4	11.13	1339.6
$\mathsf{R}=4$	9.31	164.4	9.98	3782.0	10.55	85.5	11.21	2047.0

d) Choice of redundancy: As in denoising experiments, processing times in Table VII lead us to retain only the twice redundant setting ($\mathsf{R} = 2$) for further comparisons.

e) Analysis vs synthesis: Even if we introduce both the analysis and synthesis versions of the declippers in sections IV-B1 and IV-B2, the detailed experimental results below will compare only the analysis social and analysis plain sparse declipping methods: while both flavors perform similarly qualitatively speaking, the computational cost of the synthesis declippers (See. Table VII and Section IV-B) is much higher. Indeed the analysis version allows a computational speedup of the order of 23 to 139 depending on the redundancy and the input SDR.

TABLE VIII
COMPUTATIONAL PERFORMANCE OF COSPARSE DECLIPPERS

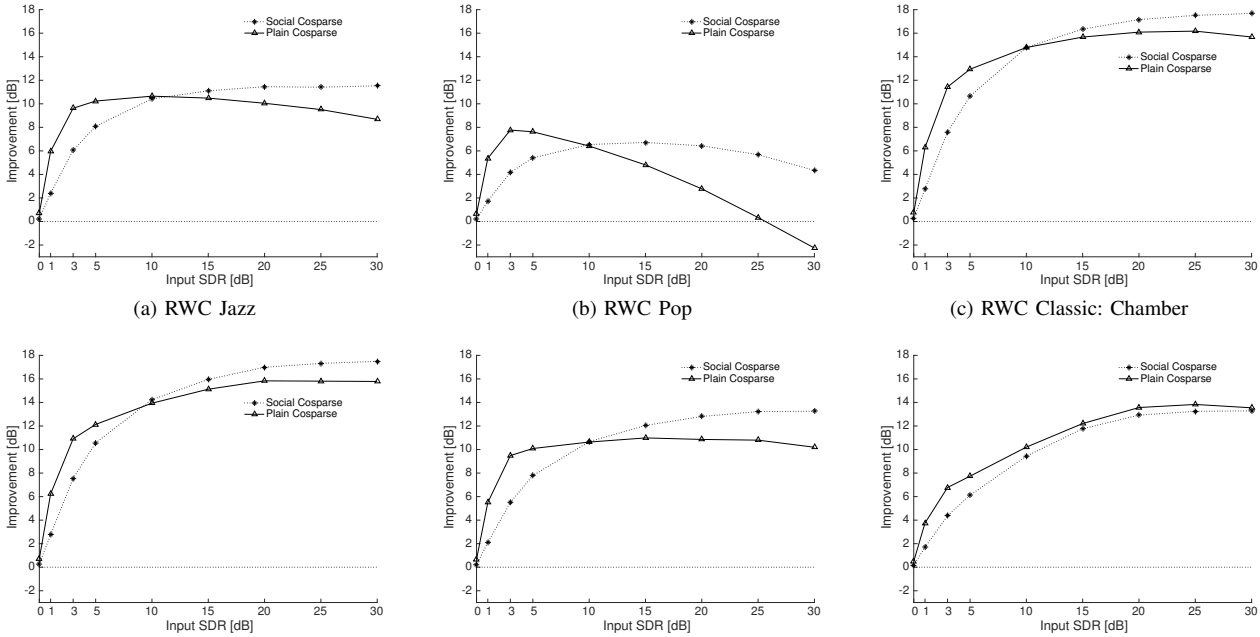
Input SDR [dB]	Plain Cosparse		Social Cosparse	
	Δ SDR	$\times$ RT	Δ SDR	$\times$ RT
0	0.64	27.4	0.20	188.1
5	9.73	41.7	7.37	130.4
10	10.55	46.4	9.70	113.0
15	10.81	37.9	11.0	84.3
20	11.04	24.4	12.31	50.2

f) Computation time: Table VIII provides processing times for both analysis social sparse (Social Cosparse) and analysis plain sparse (Plain Cosparse) declipping methods. These experiments are conducted using the same hardware and software settings described in the denoising experimental section. The cosparse declippers benefit from the same computational assets as the denoisers. We notice again that the social cosparse declipper runtime performance seems to improve as the degradation level decreases.

g) Comparison of declipping performance: To evaluate the different sparse modelings, we use the shrinkages (Hard Thresholding and PEW). We also set the clipping level of the saturated material to account for different degrees of degradation. For that, we consider nine input SDR levels in dB: {0, 1, 3, 5, 10, 15, 20, 25, 30}. Results presented on Figures 9 and 10 show averaged measurements over all available excerpts.

Figure 9 shows the behavior of the two methods as a function of the input degradation level. For all the considered datasets, both declipping methods provide significant SDR improvements (often more than 8dB) at (almost) all considered input SDRs. Unlike for denoising, this remains the case even for relatively high input SDRs, with one exception: the Pop category, for which the Plain Cosparse brings some degradation at very high input SDR, and the overall improvement never exceeds 8dB. This may be due to the fact that most of the 100 unclipped excerpts in this category are mixes containing one or more tracks of dynamically compressed drums, and that at least 21 of them contain saturated guitar sounds.

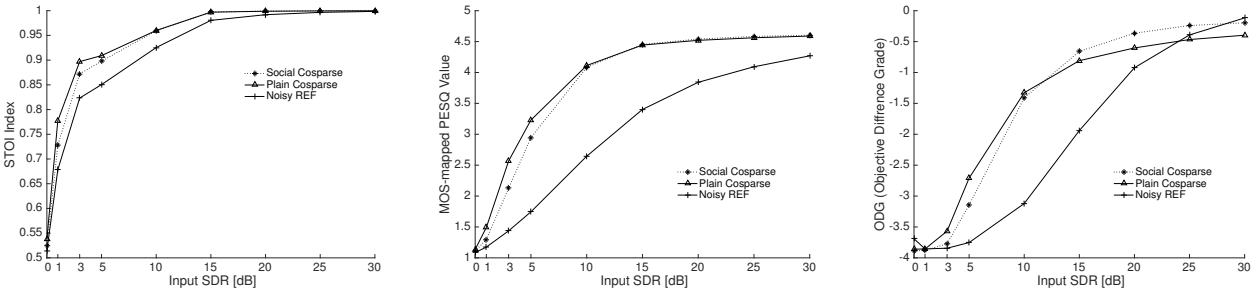
The benefit of social modeling is clear for moderate to high input SDR ($>$ 10dB, mild clipping), and vice-versa there is also a distinct superiority of the plain cosparse method for low input SDRs (strong clipping). Actually, the simple cosparse approach performs 2 to 4 dB better than the adaptive social method for input SDRs ranging from 1 to 5 dB on audio content from the RWC database. On the opposite, the trend tends to reverse above 10 dB input SDR as the social methods features improvements between 1 and 4 dB (even 7dB for the Pop category) above the plain cosparse technique. For speech content, the difference is less obvious yet Fig. 9f, Fig. 10a and 10b displays better improvements either in terms of SDR, objective intelligibility or quality for



(a) RWC Jazz (b) RWC Pop (c) RWC Classic: Chamber

(d) RWC Classic: Vocals (e) RWC Classic: Symphonies (f) TIMIT: Speech

Fig. 9. Numerical Results for the declipping task: SDR improvement [dB]



(a) TIMIT: STOI Index (b) TIMIT: MOS-Mapped PESQ Value (c) RWC: PEAQ ODG score

Fig. 10. Declipping Objective Quality and Intelligibility Results

the plain cosparse declipper.

Contrarily to denoising settings, standard deviation results indicate that the social cosparse declipper produces more consistent results as the standard deviation is the lowest for this technique in 67% of the tested cases. We also observed that, for any of the considered algorithms, the improvement variability seems to increase with the input SDR.

The difference in performance between the plain and social cosparse declippers on music at low input SDR might come from the nature of the degradation. Indeed, contrarily to additive noise, the magnitude saturation adds broadband stripes in the time-frequency plane due to discontinuities in the time domain. This way, the signal's underlying structure (embodied by a time-frequency pattern Γ) is not only hidden as in the additive noise case, but also possibly distorted: during the initialization loop of the social approach, it is possible that a "wrong" pattern Γ^* is selected. In contrast, the plain cosparse declipper cannot be affected by this type of behaviour. Another interesting result which could supports this hypothesis is that for higher SDR, the social method is actually benefiting from the time-frequency structure identification as it performs better.

V. Conclusion

The algorithmic framework for audio restoration proposed in this paper is versatile from many points of view.

First, it can handle several types of audio distortion models, as demonstrated here on two audio restoration problems, denoising and declipping. More restoration scenarios are envisioned, and further work will include in particular multichannel scenarios and extensions to non-Gaussian noise.

Second, it handles transparently both the analysis and synthesis flavors of time-frequency models. On pilot studies, the analysis and synthesis flavors yields almost identical SNR performance on denoising and declipping, the analysis version being between 20% and nearly 40% faster than the synthesis one on denoising, and more than 20 times faster on declipping. In fact, real-time was achieved on experiments not shown here. The plain and social flavors have comparable qualitative

restoration performance but reach their maximum speed in complementary input degradation regimes.

In terms of quality, our detailed study of the analysis version for declipping shows SDR improvements consistently exceeding 8dB for various types of speech and music and a wide range of clipping levels. The only notable exception is the Pop dataset, possibly due to the presence of dynamically compressed drums and saturated guitar sounds. Similarly, in the denoising scenario, consistent SNR improvements are observed that are either on par with or better than what the widely used Block Thresholding reference algorithm can achieve, especially for input SNRs above 10dB. Similar trends are observed with perceptually-aware objective quality measures, however further work would be needed to confirm them on subjective listening tests.

Finally, the framework can seemlessly exploit plain or social sparsity through the integration of appropriate shrinkage operators. For declipping, social cosparsity brings a clear benefit for moderate but still significant saturation levels (input SDR over 10dB), while plain cosparsity seems consistently preferable for strongly clipped scenarios. This is possibly due to the difficulty to properly detect social sparsity patterns heavily corrupted by strong saturation, and calls for further studies to guide the choice of such patterns possibly based on learning techniques. For denoising, the behaviour is opposite, as the performance of the plain and social flavors of the framework seem relatively similar except at low input SNR where social sparsity brings a mild benefit compared to the plain one.

Appendix A
Generalized projection for denoising

The goal is to solve (3), i.e.,

$$\underset{\mathbf{W}\in\Theta}{\text{minimize}}\ \|\mathbf{M}\mathbf{W}-\mathbf{Z}\|_{\text{F}}.$$

with $\mathbf{M}=\mathbf{A}$, $\Theta=\{\mathbf{W}:\|\mathbf{W}-\mathbf{Y}\|_{\text{F}}\leq\varepsilon\}$ for the analysis case, and $\mathbf{M}=\mathbf{I}$ and $\Theta=\{\mathbf{W}:\|\mathbf{D}\mathbf{W}-\mathbf{Y}\|_{\text{F}}\leq\varepsilon\}$ for the synthesis case. For the synthesis case, this is more explicitly

$$\underset{\mathbf{W}}{\text{minimize}}\ \|\mathbf{W}-\mathbf{Z}\|_{\text{F}}^2 \text{ subject to } \|\mathbf{D}\mathbf{W}-\mathbf{Y}\|_{\text{F}}^2\leq\varepsilon^2.$$

Let us now show that in the analysis case the optimization problem can be cast to a similar form. Since we consider a tight frame $\mathbf{A}^{\text{H}}\mathbf{A}=\zeta\mathbf{I}$, the orthogonal projection onto the linear span of $\mathbf{A}$ is $P_{\mathbf{A}}=\zeta^{-1}\mathbf{A}\mathbf{A}^{\text{H}}$ and for any $\mathbf{W},\mathbf{Z}$,

$$\begin{aligned}\|\mathbf{A}\mathbf{W}-\mathbf{Z}\|_{\text{F}}^2 &= \|\mathbf{A}\mathbf{W}-P_{\mathbf{A}}\mathbf{Z}+(\mathbf{I}-P_{\mathbf{A}})\mathbf{Z}\|_{\text{F}}^2\\ &= \|\mathbf{A}\mathbf{W}-P_{\mathbf{A}}\mathbf{Z}\|_{\text{F}}^2+\|(\mathbf{I}-P_{\mathbf{A}})\mathbf{Z}\|_{\text{F}}^2\\ &= \|\mathbf{A}(\mathbf{W}-\zeta^{-1}\mathbf{A}^{\text{H}}\mathbf{Z})\|_{\text{F}}^2+\|(\mathbf{I}-P_{\mathbf{A}})\mathbf{Z}\|_{\text{F}}^2\\ &= \zeta\|\mathbf{W}-\zeta^{-1}\mathbf{A}^{\text{H}}\mathbf{Z}\|_{\text{F}}^2+\|(\mathbf{I}-P_{\mathbf{A}})\mathbf{Z}\|_{\text{F}}^2.\end{aligned}$$

Minimizing the left hand side with the constraint $\mathbf{W}\in\Theta$ is thus equivalent to

$$\underset{\mathbf{W}}{\text{minimize}}\ \|\mathbf{W}-\zeta^{-1}\mathbf{A}^{\text{H}}\mathbf{Z}\|_{\text{F}}^2 \text{ subject to } \|\mathbf{W}-\mathbf{Y}\|_{\text{F}}^2\leq\varepsilon^2.$$

Both cases boil down to an optimization problem

$$\hat{\mathbf{W}}=\underset{\mathbf{W}}{\text{argmin}}\ \|\mathbf{W}-\mathbf{B}\|_{\text{F}}^2 \text{ subject to } \|\mathbf{F}\mathbf{W}-\mathbf{Y}\|_{\text{F}}\leq\varepsilon \quad (13)$$

with $\mathbf{B}=\zeta^{-1}\mathbf{A}^{\text{H}}\mathbf{Z}$ and $\mathbf{F}=\mathbf{I}$ for the analysis case, while $\mathbf{B}=\mathbf{Z}$ and $\mathbf{F}=\mathbf{D}$ for the synthesis case. When $\mathbf{F}$ is a tight frame, $\mathbf{F}\mathbf{F}^{\text{H}}=\xi\mathbf{I}$, problem (13) has a closed form solution [24, Section 2]

$$\hat{\mathbf{W}}=\mathbf{B}-\frac{1}{\xi}\left(\frac{\|\mathbf{F}\mathbf{B}-\mathbf{Y}\|_{\text{F}}-\varepsilon}{\|\mathbf{F}\mathbf{B}-\mathbf{Y}\|_{\text{F}}}\right)_+\cdot\mathbf{F}^{\text{H}}(\mathbf{F}\mathbf{B}-\mathbf{Y}). \quad (14)$$

Appendix B
Generalized projection for declipping

The goal is to solve (3), i.e.,

$$\underset{\mathbf{W}\in\Theta}{\text{minimize}}\ \|\mathbf{M}\mathbf{W}-\mathbf{Z}\|_{\text{F}}.$$

with some constraint set Θ.

In the analysis case, as shown in Appendix A, as soon as $\mathbf{A}^{\text{H}}\mathbf{A}=\mathbf{I}$, minimizing $\|\mathbf{A}\mathbf{W}-\mathbf{Z}\|_{\text{F}}^2$ under the constraint

$$\mathbf{W}\in\Theta:=\left\{\begin{array}{ll}\mathbf{W}\mid & \mathbf{W}_{(\text{ij})}=\mathbf{Y}_{(\text{ij})},\ \text{ij}\in\Omega_r;\\ & \mathbf{W}_{(\text{ij})}\geq\tau,\text{ij}\in\Omega_+;\\ & \mathbf{W}_{(\text{ij})}\leq-\tau,\ \text{ij}\in\Omega_-.\end{array}\right\}$$

is equivalent to minimizing $\|\mathbf{W}-\mathbf{A}^{\text{H}}\mathbf{Z}\|_{\text{F}}^2$ under the constraint $\mathbf{W}\in\Theta$. As the contraint is written componentwise, the optimization can be done componentwise yielding

$$\hat{\mathbf{W}}_{(\text{ij})}=\left\{\begin{array}{ll}\mathbf{Y}_{(\text{ij})} & \text{if ij}\in\Omega_r;\\ (\mathbf{A}^{\text{H}}\mathbf{Z})_{(\text{ij})} & \text{if}\left\{\begin{array}{l}\text{ij}\in\Omega_+,(\mathbf{A}^{\text{H}}\mathbf{Z})_{(\text{ij})}\geq\tau;\\ \text{or}\\ \text{ij}\in\Omega_-,(\mathbf{A}^{\text{H}}\mathbf{Z})_{(\text{ij})}\leq-\tau;\end{array}\right.\\ \text{sgn}\,(\mathbf{Y}_{\text{ij}})\tau & \text{otherwise.}\end{array}\right. \quad (15)$$

For the synthesis case, $\mathbf{M}=\mathbf{I}$ and

$$\Theta:=\left\{\begin{array}{ll}\mathbf{W}\mid & (\mathbf{D}\mathbf{W})_{\Omega_r}=\mathbf{Y}_{\Omega_r};\\ & (\mathbf{D}\mathbf{W})_{\Omega_+}\succcurlyeq\mathbf{Y}_{\Omega_+};\\ & (\mathbf{D}\mathbf{W})_{\Omega_-}\preccurlyeq\mathbf{Y}_{\Omega_-}.\end{array}\right\}.$$

The corresponding optimization problem for the synthesis case writes:

$$\hat{\mathbf{W}}=\underset{\mathbf{W}}{\text{argmin}}\ \|\mathbf{W}-\mathbf{Z}\|_{\text{F}}^2 \text{ subject to } \mathbf{D}\mathbf{W}\in\Theta. \quad (16)$$

Unfortunately, as the magnitude constraint is expressed here in the frequency domain, there is no simple expression for $\hat{\mathbf{W}}$. As explained in [6], we can solve (16) with an iterative procedure, leading to the much higher cost of the synthesis (non)social sparse declipper.

Acknowledgment

This work benefits from the financial support of the European Research Council, PLEASE project (ERC-StG-2011-277906), and of the Brittany Region (ARED 9173). The authors would like to thank Matthieu Kowalski for precious discussion and advice.

REFERENCES

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, Apr 1979.

[2] R. McAulay and M. Malpass, "Speech enhancement using a soft-decision noise suppression filter," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 2, pp. 137–145, Apr 1980.

[3] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[4] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[5] G. Yu, S. Mallat, and E. Bacry, "Audio denoising by time-frequency block thresholding," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1830–1839, 2008.

[6] S. Kitić, "Cosparse regularization of physics-driven inverse problems," PhD Thesis, IRISA, Inria Rennes, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/tel-01237323

[7] S. Kitić, N. Bertin, and R. Gribonval, "Sparsity and cosparsity for audio declipping: a flexible non-convex approach," in *Latent Variable Analysis and Signal Separation (LVA/ICA)*. Liberec, Czech Republic: Springer, 2015, pp. 243–250.

[8] R. Jenatton, J.-Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *The Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[9] M. Kowalski, K. Siedenburg, and M. Dorfler, "Social sparsity! neighborhood systems enrich structured shrinkage operators," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2498–2511, 2013.

[10] J. Eckstein and D. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[11] M. Kowalski, "Thresholding rules and iterative shrinkage/thresholding algorithm: A convergence study," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 4151–4155.

[12] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[13] K. Siedenburg, M. Kowalski, and M. Dorfler, "Audio declipping with social sparsity," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1577–1581.

[14] K. Siedenburg and M. Dörfler, "Audio denoising by generalized time-frequency thresholding," in *Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio*. Audio Engineering Society, 2012.

[15] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[16] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases." in *ISMIR*, vol. 2, 2002, pp. 287–288.

[17] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[18] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *Proceedings .(ICASSP'01). IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 2001, pp. 749–752.

[19] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[20] P. Kabal, "An examination and interpretation of itu-r bs. 1387: Perceptual evaluation of audio quality," *TSP Lab Technical Report, Dept. Electrical & Computer Engineering, McGill University*, pp. 1–89, 2002.

[21] Y. Tachioka, T. Narita, and J. Ishii, "Speech recognition performance estimation for clipped speech based on objective measures," *Acoustical Science and Technology*, vol. 35, no. 6, pp. 324–326, 2014.

[22] M. J. Harvilla and R. M. Stern, "Least squares signal declipping for robust speech recognition," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[23] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in Blind Audio Source Separation," *IEEE Trans. Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[24] J. Yang and X. Yuan, "Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization," *Mathematics of computation*, vol. 82, no. 281, pp. 301–329, 2013.